整合性維持に着目したソースコードとドキュメントの一元管理環境の提案

赤石 裕里花　坂井 麻里恵　奥野 拓　伊藤 恵

ソフトウェア開発においてドキュメントは，ソースコードの正しい理解を促進し，ソフトウェアの品質を保つ意味で重要なものである．最新の信頼できる情報を保つために，ドキュメントの管理が重要だ．しかし，各工程の成果物として作られる多種多量のドキュメントは作成後の管理が煩雑になり，ソフトウェアの仕様変更があった際に，ドキュメントの変更漏れが生じやすい．その結果，ドキュメントの説明とソースコードの動作が一致しないことが発生する．この問題を解決するために，本研究ではソースコード内の自然言語によるコメント等を活用した一元管理環境の提案をする．これによりソースコードとドキュメントの整合性を図る．

It is important that software documents encourage the proper understanding of source code, and maintain quality of the software. It is important to manage software documents in order to keep the most current and credible information. But, after developer create various software documents, these management become complex. If the software specifications change,the changes of software documents tend to miss. Consequently, behavior expressed in source code and its description in software document may differ. In this study, we propose an unified management environment of source code and software documents, utilizing natural language comments in source code. This environment is aiming that maintain consistency between the information of source code and software documents.

1　はじめに

1.1　背景

ソフトウェア開発では，各工程の成果物として様々なドキュメントが作られる．ドキュメントは開発工程によって作成される意味合いが異なり，コーディング前はまだ動くものが作られていない場合が多いため，作るイメージを具体化し，ステークホルダ間で共有するために作られる．コーディング後は，ソースコードを理解するために使われたり，マニュアルのように完成したソフトウェアを利用方法を知るために使われたりする．このようにドキュメントはソフトウェア開発の中で重要な成果物となる．

共通フレーム 2007 [2] から，主ライフサイクルプロセス中の 8 つのプロセスで作られるドキュメントの種類をあげる．ドキュメントは共通フレームの中で「文書化」しているものを指すこととし,「記述」している部分に関しては，前のプロセスまでに作られたドキュメントの中に追加で記述するものとしたためここでは除く．

取得プロセスでは提案依頼書が，供給プロセスでは提案書や計画書が主に作られ，一番ドキュメント数が多い開発プロセスでは，ソフトウェア詳細設計，ソフトウェアコード，テスト計画などおよそ 20 種のドキュメントが作られる．

開発プロセスでは，詳細設計に基づきコーディングを行い，ソースコードを作成するが，のちの運用プロセスや保守プロセスで，ソフトウェアの仕様修正・追加する場合にドキュメントとソースコードの整合性が取れなくなってしまうことがある．

以下にソースコードとドキュメントの整合が取れな

Proposal of Management Environment of Source Code and Software Documents for Consistency Maintenance

Yurika Akaishi, Taku Okuno, Kei Ito, 公立はこだて未来大学システム情報科学部, Dept. of Systems Information Science, Future University Hakodate.

Marie Sakai, 公立はこだて未来大学大学院システム情報科学研究科, Graduate School of Systems Information Science, Future University Hakodate.

い原因を挙げる．

- ソースコード中に参照すべきドキュメントについて書かれていない
- 参照すべきソースコードとドキュメントの関係が多対多である
- ドキュメントの数，種類が多種多量である
- 2 種類以上のドキュメントに重複した記述がある

これらの原因により，ソースコードにしたがってドキュメントを変更しようとしても，変更すべき箇所を探すのが煩雑であったり，漏れが生じたりして，ドキュメントが正しく更新されない．

1.2 目的

本研究では「ソースコード中に参照すべきドキュメントについて書かれていない」点に着目し，ソースコードとドキュメントの一元管理環境を提案することを目的とする．本研究の提案する一元管理環境は，ソースコードとドキュメントの整合性維持を実現させ，ドキュメントの変更点を探す煩雑さを解消する．

また，ソースコードとドキュメント自体の可読性も失わせないことを目指し，各自のファイルの見易さも重視する．

2 関連研究

2.1 文芸的プログラミング

文芸的プログラミング (Liteate Programing) [6] とは，Donald Knuth が提唱したプログラミングの手法である．

Donald Knuth はプログラミングには 3 重の意義があると述べている．(美的な面，人道的な面，財政的な面) そしてこの 3 通りの意義を達成するプログラミングは文学的であると考え,「コンピュータプログラムは機械が実行可能でなければならないが，それは主な目的ではない．真に美しく，有用な，そして収入を得られるコンピュータプログラムは，人間が読むことができなければならない」という考えの下文芸的プログラミングは成り立っている．

ソースコードとドキュメントをひとつのファイルに混在させながら開発していくことにより，両者の整合性をとることを目指している．文芸的プログラミング

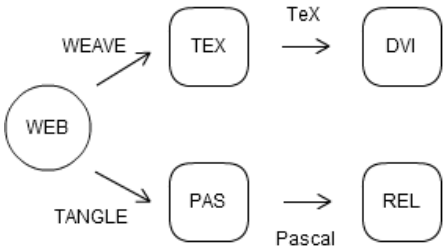

図 1　WEB システムの構造

によって書かれたソースコードは，ドキュメント部分が多くを占めるので，従来のソースコードと大きく異なる．Donald Knuth が自身の研究で WEB システムと名付けている．その WEB システムの構造を図 1 に示す．

2 種類の異なる言語でコーディングを行い，ソースコードを WEB ファイルとして格納する．WEAVE の処理はソースコードのドキュメント部分を生成し，TANGLE の処理では機械実行が可能なプログラムを生成する．WEB から生成されるドキュメントとプログラムは，同じ WEB ファイルから生成されるため，整合が取れている．なお，図中では TeX 言語と Pascal 言語を使用しているが，WEB の原理は他の言語を用いても同様に適用できる．

2.2 ADIOS

ADIOS [3] はソフトウェアに関係する多種のドキュメントをアスペクト指向を用いて整理するドキュメント整理ツールである．開発工程で発生する様々な種類のドキュメントとソースコードの対応付けを行う．ドキュメントの種類やキーワードを関連と呼び，この関連をソースコード内に一定の形式で埋め込む．関連を埋め込む作業は手動で行うが，その後は ADIOS によって埋め込まれた関連を収集し管理することができる．図 2 は ADIOS の実行画面である．

2.3 Prio

Prio [4] は，仕様書の不整合とソースコードの不整合を自動的に検出するツールである．XML 言語で書かれた仕様書の文字列にソースコード内の識別子と意

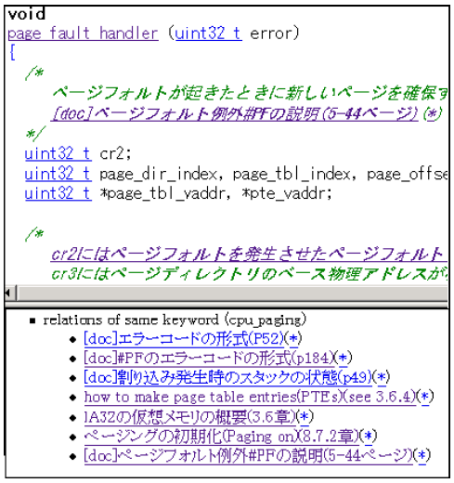

図 2　ADIOS の実行画面

味付けを行い，条件が抜けている場合と，記述が抜けている場合を不整合と判別し，その部分が強調表示される．Prio はエディタ部とチェッカ部から構成されており，エディタ部でテキストエディタに似た操作でプログラムの断片情報を仕様書の仕様断片に埋め込み，チェッカ部で仕様断片とプログラム断片間の設定条件に含まれる不整合部分を自動的に抽出する．

2.4　A HotDocument

A HotDocument は，様々な言語のソースコードをドキュメントとして自動生成するツールである．出力するドキュメントの種類も Excel/テキスト/HTML/CHM/XML の形式を選択できる．A HotDocument から生成されるドキュメントの内容は，大きく 2 種類に分かれている．1 つはソースコードの実コードを解析してドキュメント化する部分，もう 1 つはソースコードのコメントを抜き出してドキュメント化する部分である．そのため，A HotDocument のコメント規約に従っていないソースコードは，説明の項目だけが空欄になり，それ以外はすべてドキュメント化できる．

2.5　ソースコードのコメントを活用したドキュメント生成ツール

コメント規約に基づきソースコードにコメントを記述し，そのコメントからドキュメントを生成するツールを紹介する．

(1) Javadoc

Javadoc は JDK に標準で含まれているプログラムで，ソースコードに含まれるクラスやメソッドのコメントを Javadoc のコマンドを用いて自動的に HTML 形式のドキュメントとして出力する．また，関連するドキュメントの URI を埋め込むこともできる．主にクラスの概要やメソッドの概要を記述し，それをまとめたドキュメントを得るために使われる．

(2) Doxygen

Javadoc と同様にソースコードからドキュメントを生成するツールであるが，Javadoc は Java に対応しているのに対し，Doxygen では Java 言語の他にも多くの言語に対応しており，出力するドキュメントも HTML，LaTeX，RTF (MS-Word)，PostScript，ハイパーリンク PDF，圧縮 HTML，Unix man ページ形式の出力もサポートされている．

これらは，コメントの記述形式が限定されているので，形式にしたがって記述する必要がある．また，出力するドキュメントを生成するためにソースコード内にコメントを書くため，もともと書かれたドキュメントの内容と重複した内容を書かなければならない．作成後の管理も重複して行うことになる．

3　提案

本研究では，開発プロセスの中の詳細設計書作成，コーディング，その後のソースコードとドキュメントの整合性を管理する環境として，文芸的プログラミングの思想と関連研究の手法を取り入れ，ソースコードとドキュメントを混在して閲覧・編集する手法を提案する．これによりドキュメントとソースコードの整合性が維持できる．また，可読性の部分に関しても考慮し，ドキュメント部分とソースコード部分を別々に出力する．

そして運用プロセスや保守プロセスで仕様変更があった場合にも，煩わしい動作をせずソースコードとそれに対応するドキュメントの変更・更新を促す．提案環境は閲覧ツール (Viewer)，編集ツール (Editor)，変換ツール (Converter) の 3 つのツールからなる．それぞれのツールの詳細については 3.1 節のツールの詳細に後述する．対象とするドキュメントは，コーディングの際に参考にする詳細設計書とプログラムの動作を説明するドキュメントを対象とする．ドキュメントは XML 形式で保存することとし，出力する際はドキュメントとして見やすいように変換される．

3.1 提案するツールの詳細

図 3 に提案ツールの概観を示す．まず開発者が XML 形式のドキュメントを作成する．そのドキュメントは開発プロセスでの詳細設計にあたる．そのドキュメントにはソースコードに準ずる情報が書かれているので，そのドキュメントにしたがってコーディングを行い，ソースコードを作成する．

ソースコードとドキュメントの作成後，後から見直したい場合，Viewer を用いてソースコードとドキュメントの閲覧を行う．別ファイルで保存されているソースコードとドキュメントを混在した形で表示させる．仕様変更があった場合は，Editor を用いてソースコードとドキュメントの編集を行う．Viewer のようにソースコードとドキュメントが混在し表示されており，お互いの関連する箇所が近い場所に表示されるようになっている．このため，ソースコードの変更後，それに従って近くに表示されている変更箇所に関係するドキュメントの部分を変更する．

Converter は，Editor で出力する際に用い，自動的にソースコードとドキュメントを適切な形式で保存する．具体的には，ソースコードはコンパイルを行い実行可能なファイルに変換し，XML 形式のドキュメントはドキュメントの部分だけを抜き取り実際の詳細設計書のように体裁の整ったファイルに変換する．設計書に書かれていることのみを参照したい場合は，ドキュメントの部分だけを抜き出したファイルを参照し，逆にプログラムを実行させたいときは，実行ファイルから実行すればよい．

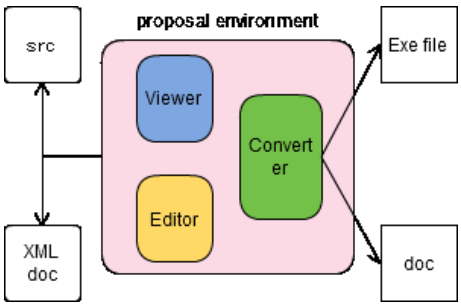

図 3　提案環境の概観

図 4　Javadoc により生成された API 仕様書

3.2 ソースコードとドキュメントの対応付け例

提案するツールのイメージとして，Javadoc で簡単なプログラムのソースコード [1] と，関連する HTML 形式のドキュメントを生成した．Javadoc を採用した理由は，提案ツールでは編集のしやすさと整合性の観点から、ソースコード内にドキュメントの内容が混在している点と，ソースコードからドキュメントを生成することで，生成時には整合性が取れている点より採用した．Javadoc のコメントを記述したソースコードをソースコード 1 に示す．

ソースコード 1　sample01.java

```
1  /**
2   * Javadocテスト用クラス
3   * @author JavaDrive
4   * @version 1.0
5   */
6
7  /**
8   * Javadocテスト用クラス
9   * @author JavaDrive
10  * @version 1.0
```

```
11  */
12 public class Sample01{
13
14   /** 幅 */
15   private int w;
16
17   /** 高さ */
18   private int h;
19
20   /** デフォルトコンストラクタ */
21   Sample01(){
22     w = 0;
23     h = 0;
24   }
25
26   /**
27    * サイズの設定
28    * @param width 幅
29    * @param height 高さ
30    */
31   public void setSize(int width, int
       height){
32     w = width;
33     h = height;
34   }
35
36   /**
37    * 幅の取得
38    * @return 幅
39    */
40   public int getWidth(){
41     return w;
42   }
43
44   /**
45    * 高さの取得
46    * @return 高さ
47    */
48   public int getHeight(){
49     return h;
50   }
51 }
```

ソースコード内のコメントは Javadoc のコメント規約で記述されている．/** ～ */で囲まれている部分が Javadoc のコメントである．また頭に”@” 記号が付くコメントは，Javadoc のタグで，ある一定の記述内容が決められており，生成された HTML に含まれる．

このソースコードから，HTML 形式のドキュメントを生成した．図 3 に示す．Javadoc で生成されるドキュメントは API 仕様書として利用されている．本研究では整合性維持を目的としているため，

3.3 提案ツールの評価

一元管理環境のツールを実装し，総務省の「最先端ネットワーク技術を活用した遠隔教育システムの開発・実証」[5] の PBL 学習教材を用いてツールの評価を行う．教材中のソースコードを用いて，提案ツールを使用しソースコードの変更に伴うドキュメントの変更を行い，両者に矛盾点がないか調べる．ソースコードとドキュメントの矛盾点がなかった場合，整合性が取れているといえる．その他にも，提案ツールを用いず変更した場合や既存ツールを用いた場合と比較し，かかった時間やドキュメントの有用性も評価の対象とする．評価項目は以下の通りとする．

(1) ソースコードとドキュメントの整合性
(2) ソースコードの改変量
(3) (ドキュメント自動生成型の既存ツールを用いた場合) 生成したドキュメントの有用性

4 考察

本研究で提案したソースコードとドキュメントの一元管理環境で開発を行うことにより，従来の開発環境よりも，ソースコードとドキュメントのリアルタイムでの更新が可能になり，整合性が取りやすくなると考える．以下に提案環境のツールから得られる可能性のある結果を考察する．

(1) Viewer
別ファイルで保存されているソースコードとドキュメントが混在させた形で見えるため，従来のソースコードからプログラムの動作を読み解くというよりは，自然言語のドキュメントを読みながらソースコードで動作を確認するという感覚になる．クラスやメソッドの細かい情報を別ファイルのドキュメントから探すことなく，ソースコードを読むことができるのためソースコードの理解が早まると考える．

(2) Editor
Viewer と同様の表示方法で，編集することができるため，ソースコードとドキュメントを同時に修正できる．ソースコードとドキュメントを別々

に表示させる方法と比べると，変更点を探す時間が短縮され，変更する際も不整合が起こりにくくなると考える．

(3) Converter

このツールは，自動的に起動し実行され，ツールの利用者が直接的に操作することはないため，ここでは言及しない．

5 まとめと今後の課題

ソフトウェア開発で作成される，ソースコードとドキュメントの一元管理環境の提案をした．ソースコードと XML で記述されたドキュメントを混在させ表示させる Viewer と，編集する場合は Viewer 同様の表示方法で編集する Editor によって構成されている．これにより，わざわざ編集箇所を探す手間が省け，ソースコードとドキュメントを同時に編集できる．この結果整合性が取りやすくなる．

また，Javadoc を用いて提案ツールに近いプロタイプを試みた．Javadoc ではソースコードとドキュメントが同一のファイルに保存されているが，提案ツールでは別々のファイルを混在して表示する方法を工夫すべきだと考える．この他にも今後の課題として，XML ドキュメントに記述する内容，ソースコードと XML ドキュメントの対応付けをどのように行うのか，ソースコードとドキュメントを混在させた表示方法，提案ツールの評価の詳細決定を行う必要がある．

参 考 文 献

[1] IKURA, T.: ドキュメントの作成. http://www.javadrive.jp/javadoc/ini/index3.html.

[2] 独立行政法人 情報処理推進機構ソフトウェア・エンジニアリング・センター: 共通フレーム *2007*, 株式会社オーム社, 2007.

[3] 大場勝, 権藤克彦: アスペクト指向を用いたドキュメント整理法の提案, 日本ソフトウェア科学会第 7 回プログラミングおよび応用のシステムに関するワークショップ, (2004).

[4] 山田信幸, 鈴木幹雄, 坂井守: XML 形式の仕様書作成によるソフトウェア机上チェックの効率化 - 机上デバッグ支援ツール Prio （プライオ）- , ソフトウェアテストシンポジウム *2003*, (2003).

[5] 総務省: PBL 学習（システム開発プロジェクト）, http://www.soumu.go.jp/main_sosiki/joho_tsusin/joho_jinzai/, 2013.

[6] Donald E. Knuth 有澤 誠訳: 文芸的プログラミング, アスキー出版局, 1994.